Ludwig-Maximilians-Universität München
LFE Medieninformatik
Prof. Dr. Andreas Butz
Dipl. Inf. Otmar Hilliges

Sommersemester 2007
Übungsblatt 2
2.Mai 2007

3D Programmierpraktikum

Abgabetermin:
Die Lösung zu diesem Übungsblatt ist bis zum Montag den **7. Mai 2007, 12:00 Uhr s.t.** abzugeben.

Inhalt:
Dieses Übungsblatt behandelt die Verwendung von Klassen, Objekten und Funktionen. Außerdem wird das Konzept der Vererbung und insbesondere die Mehrfachvererbung in C++ betrachtet.

Aufgabe 4 (P) Klassen, Mehrfachvererbung

In dieser Aufgabe erlernen sie den Umgang mit Klassen, Header-Dateien sowie Vererbung und Mehrfachvererbung. Dazu dient ein Beispiel aus der Zoologie.

a) Betrachten sie die Tiere *Löwe*, *Haifisch* und *Krokodil*. Alle drei haben sowohl gemeinsame als auch unterschiedliche Eigenschaften. Zu erst sind alles drei Tiere, allerdings lassen sich die drei Kandidaten noch weiter unterteilen. Teilen Sie die Tiere in *Landtiere* und *Wassertiere* ein. Überlegen sie sich, um was es sich bei einem Krokodil handelt. Bedenken sie, dass in *C++* auch mehrfache Vererbung möglich ist und berücksichtigen sie dies in ihren Überlegungen. Erstellen sie nun ein UML-Diagramm, das die Zusammenhänge und Unterschiede herausstellt.

b) Erweitern Sie ihr Design an geeigneten Stellen um folgende Eigenschaften der Tiere:

- Tiere besitzen ein Alter (Ganzzahlwert)
- Tiere können sich fortbewegen.
- Tiere haben im laufe Ihres Lebens eine gewisse Entfernung zurückgelegt (diese kann im Wasser anders sein als am Land)
- Ein *Haifisch* kann eine bestimmte Distanz schwimmen
- Ein *Krokodil* kann eine bestimmte Distanz schwimmen oder laufen
- Ein *Löwe* kann eine bestimmte Distanz laufen

c) Erstellen Sie nun Header-Dateien für jede Klasse, die in Ihrem UML-Diagramm erscheint. Das Diagramm sollte die drei Tiere, die zwei verschiedenen Aufenthaltsklassifikationen (Land oder Wasser) sowie eine allgemeine Klasse für die gemeinsamen Eigenschaften enthalten. Geben Sie Ihren Klassen ferner die Möglichkeit ihren Status aus zu geben. Erstellen Sie zudem alle Klassen im Namensraum `Tiere`.

Beispiel: Mögliche Ausgabe für ein Krokodil:
`Das Krokodil (4 Jahre alt) schwamm 12 sm und lief 10 km.`

d) Implementieren Sie nun diese Klassen. Achten Sie dabei auf eventuell zu überschreibende Funktionen und verwenden Sie das Schlüsselwort `virtual` um den Aufruf der falschen Funktion (also eine der Superklasse) zu verhindern.

e) Erweitern Sie die Klasse *Tiere* um die Funktion `void setAge(int)`. Achten Sie jedoch darauf, dass das Alter nie negativ sein darf. Wird dennoch ein negatives Alter übergeben soll eine selbstdefinierte Ausnahme (z.B. `FalschesAlter`) erzeugt werden, die den Fehlertext enthält. Ein Beispiel für einen solchen Fehlertext wäre ”Alter darf nicht negativ sein!”.

f) Schreiben Sie schließlich die Methode `main` in der Sie nacheinander folgendes durchführen sollen:

- Als erstes Statement erzeugen Sie sich einen Zeiger auf ein Tier, z.B. `Tiere::Tier* ein_tier;`. Diesem Pointer werden Sie die drei nachfolgend erzeugten Tiere zuweisen.
- Erzeugen Sie sich einen *Haifisch* der 5 Jahre alt ist, bewegen Sie diesen um 5 Seemeilen und lassen Sie sich das Tier ausgeben
- Erzeugen Sie einen *Löwen* der 3 Jahre alt ist, bewegen Sie diesen erst um 2, dann nochmal um 5 Kilometer und lassen Sie sich das Tier ausgeben
- Erzeugen Sie ein *Krokodil* das 7 Jahre alt ist. Lassen Sie es zunächst 2 Seemeilen schwimmen, dann 5 Kilometer laufen und schließlich wieder 10 Seemeilen schwimmen. Lassen Sie sich das Tier anschließend ausgeben

g) Setzen Sie das Alter des Krokodils auf einen negativen Wert. Ändern Sie dafür außerdem die Methode `main` so ab, dass dieser inkorrekte Fall mittels eines `try-catch`-Blocks abgefangen wird. Geben Sie außerdem die Fehlermeldung (enthalten in der Ausnahme) aus.

Aufgabe 5 (P) Verwendung von Objekten, Typ-System, Typ-Umwandlung

Da Tiere Lebewesen sind, müssen sie auch Nahrung aufnehmen. In der Tierwelt gibt es unterschiedliche Ernährungsstrategien, z.B.: fleischfressende Raubtiere oder Vegetarier.

a) Erweitern Sie Ihre Klassenhierarchie an geeigneter Stelle um die Superklassen *Pflanze*, *Predator* und *Prey* (engl. für Räuber und Beute).

b) Fügen sie ihrem kleinen Ökosystem noch die Beutetiere *Antilope* und *Robbe* hinzu (Der Einfachheit halber sollen Ihre Robben Vegetarier sein, nur im Wasser leben und auch in Frischwasser heimisch sein).

c) Erweitern sie Ihr Design an geeigneter Stelle um die Methode `fressen(arg)`. Achten sie darauf, dass sie so wenige Klassen wie möglich ändern müssen. Achten sie auch darauf, dass Räuber nur Beutetiere fressen können (nicht jedoch anders herum). Achten sie auch darauf, dass Beutetiere Vegetarier sind und nur Pflanzen fressen können.

d) erweitern sie die Klasse Tier um die Membervariable `bool m_isHungry` und eine Methode um abzufragen ob ein Tier hungrig ist oder nicht.

Zu guter letzt sollen sie noch darauf achten, dass Ihre Landtiere nur Beute erreichen kann die ebenfalls an Land lebt. Das gleiche gilt natürlich für Wassertiere. Krokodile hingegen können sowohl Land- als auch Wassertiere verzehren. Hinweis: Diese Entscheidung lässt sich nur zur Laufzeit treffen verwenden sie dazu *dynamisches Casting*. Schreiben sie ein Testprogramm, welches folgende Aktionen ausführt:

a) Erzeugen sie mindestens eine Instanz der Raubtiere.

b) Erzeugen sie genug Beutetiere für alle Räuber.

c) Erzeugen sie ebenfalls genug Pflanzen für ihre Vegetarier.

d) Geben sie von allen Tieren aus, ob diese Hungrig sind oder nicht.

e) Lassen sie zunächst alle Vegetarier fressen (versuchen sie dabei auch mindestens einmal ein Raubtier zu fressen).

f) Lassen sie nacheinander alle Raubtiere fressen dabei sollen die Opfer zufällig ausgewählt werden. Achten sie darauf das gefressene Tiere nicht weiter existieren.

g) lassen sie am Ende alle Raubtiere ihren kompletten Status ausgeben (Alter, zurückgelegte Distanz, Hunger).